Foreman - Bug #10717

API host report query runs very slowly, has unnecessary clauses

06/05/2015 11:23 AM - Bryan Kearney

Status:	Resolved		
Priority:	Normal		
Assignee:			
Category:	Performance		
Target version:			
Difficulty:		**Fixed in Releases:**	
Triaged:		**Found in Releases:**	
Bugzilla link:	1228012	**Red Hat JIRA:**	
Pull request:			

Description

Cloned from https://bugzilla.redhat.com/show_bug.cgi?id=1228012
Description of problem:

Use of the hosts report in the Satellite 6 API is very slow. This is due to a query having a number of joins which are unnecessary.

Version-Release number of selected component (if applicable):

Satellite 6.0.8

How reproducible:

Always

Steps to Reproduce:
1. Request https://satellite6.example.com/api/v2/hosts/satellite6.example.com/reports?per_page=120
2. Observe resulting query.

Actual results:

Query generated is:

SELECT DISTINCT "reports".id, "reports"."reported_at" AS alias_0
FROM "reports"
LEFT OUTER JOIN "logs" ON "logs"."report_id" = "reports"."id"
LEFT OUTER JOIN "sources" ON "sources"."id" = "logs"."source_id"
LEFT OUTER JOIN "messages" ON "messages"."id" ="logs"."message_id"
LEFT OUTER JOIN "hosts" ON "hosts"."id" = "reports"."host_id" AND "hosts"."type" IN ('Host::Managed')
WHERE (("hosts"."name" = 'satellite6.example.com'))
ORDER BY "reports"."reported_at" DESC NULLS LAST LIMIT 20 OFFSET 0

As far as I can see, the logs, sources and messages tables are never used in the reporting of the query, and since they're left outer joins they do not restrict how records are joined.

The query plan of this on a customer's Satellite server was:

```
QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------
 Limit  (cost=2242850.16..2242850.21 rows=20 width=12)
   ->  Sort  (cost=2242850.16..2242891.78 rows=16646 width=12)
         Sort Key: reports.reported_at
         ->  HashAggregate  (cost=2242240.76..2242407.22 rows=16646 width=12)
               ->  Hash Left Join  (cost=1980351.18..2242157.53 rows=16646 width=12)
                     Hash Cond: (logs.source_id = sources.id)
                     ->  Hash Left Join  (cost=1978804.61..2240033.66 rows=16646 width=16)
                           Hash Cond: (logs.message_id = messages.id)
                           ->  Hash Left Join (cost=1965520.46..2226029.20 rows=16646 width=20)
                                 Hash Cond: (reports.id =logs.report_id)
                                 ->  Nested Loop  (cost=14.72..1135.03 rows=278 width=12)
                                       ->  Index Scan using index_hosts_on_name on hosts  (cost=0.
00..8.28 rows=1 width=4)
                                             Index Cond: ((name)::text ='satellite6.example.com'::
text)
                                             Filter: ((type)::text ='Host::Managed'::text)
                                       ->  Bitmap Heap Scan on reports (cost=14.72..1123.16 rows=2
87 width=16)
                                             Recheck Cond: (reports.host_id = hosts.id)
                                             ->  Bitmap Index Scan on index_reports_on_host_id  (c
ost=0.00..14.64 rows=287 width=0)
                                                   Index Cond: (reports.host_id = hosts.id)
                           ->  Hash  (cost=1107803.44..1107803.44 rows=49341944 width=12)
                           ->  Seq Scan on logs (cost=0.00..1107803.44 rows=49341944 width=12)
                      ->  Hash  (cost=12308.29..12308.29 rows=59429 width=4)
                      ->  Seq Scan on messages (cost=0.00..12308.29 rows=59429 width=4)
                ->  Hash  (cost=1037.03..1037.03 rows=31003 width=4)
                ->  Seq Scan on sources  (cost=0.00..1037.03 rows=31003 width=4)
(24 rows)
```

I believe a better query would be:

```
SELECT  DISTINCT "reports".id, "reports"."reported_at" AS alias_0
FROM "reports"
JOIN "hosts" ON "hosts"."id" = "reports"."host_id" AND "hosts"."type" IN ('Host::Managed')
WHERE (("hosts"."name" = 'satellite6.example.com'))
ORDER BY "reports"."reported_at" DESC NULLS LAST  LIMIT 20 OFFSET 0
```

There's also a sequence scan on sources, which may be able to be optimised with an index. That's a low cost operation though, compared to the other hash joins.

Expected results:

Query only joins with tables necessary for output or selection, and uses inner joins to prevent overmatching with nulls.

Additional info:

History

#1 - 06/05/2015 11:24 AM - Bryan Kearney

- Category set to Reporting

#2 - 06/08/2015 03:21 AM - Dominic Cleal

- Category changed from Reporting to Performance

#3 - 11/09/2016 11:18 AM - Chris Duryee

- Status changed from New to Closed

bug appears fixed in katello 3.0, likely due to host unification. marking as closed.

#4 - 11/10/2016 03:04 AM - Dominic Cleal

- Status changed from Closed to Resolved